AUTOMATIC MODELING OF PRONUNCIATION VARIATIONS

Ellen Eide
IBM T.J. Watson Research Center
P.O. Box 218 Room 23-126C
Yorktown Heights, NY 10598 USA

INTRODUCTION

Typically in a large vocabulary speech recognition system a word is represented as a sequence of its constituent phonemes as defined by a hand-generated lexicon, or as several alternative sequences if multiple pronunciations of the word are allowed.

A major factor in the performance of a speech recognizer is the consistency between the way the speaker pronounces the words and the way that the pronunciation is specified in the dictionary; when there is a mismatch recognition errors are likely to occur.

One approach to bridging the gap between lexicon and pronounced speech is to alter the dictionary to reflect common pronunciation variations. Alternatively, one may choose to leave the dictionary unchanged and reflect the pronuciation deviations in the acoustic model topology. In this paper we report on an implementation of the latter, an automatic method for discovering an appropriate model for each context-dependent phoneme which allows for such phenomena as reduced pronunciations and substituted phonemes where warranted by observation on training data.

During recognition in our baseline system, a hypothesized word is first expanded into its phonemic representation; each phoneme is then mapped into a sequence of subphonemic units, each of which has a simple model topology. Specifically, each phoneme maps to a sequence of three subphonemic units, each of which is modeled as a single state with a self loop. The work presented in this paper may be conceived of as replacing the mapping from phoneme to subphonemic units to simple model topology with a more complex mapping from context-dependent phoneme to an arbitrary network of states.

We show a reduction in the word error rate on both the Wall Street Journal and Broadcast News databases, with a large part of the gain on the latter coming from the spontaneous broadcast speech (F1) condition.

GENERATING OBSERVATIONS OF STRINGS

For the purposes of discussion, the processing required to build context-dependent pronunciation networks may be divided into two parts: a sequence of "pre-processing" steps resulting in sets of observations from which to train the pronunciation networks and the actual training of those networks. The material described in this section constitutes the preprocessing steps; the discussion in the next section describes the building of the pronunciation networks given the observations.

The observations from which the networks are built are sequences of label strings; each observed label sequence corresponds to the output of a constrained phoneme recognizer for the portion of the waveform aligned to a given phoneme by a Viterbi alignment.

The first step towards the generation of the sets of observation label strings from which each pronunciation network is built, then, is to perform a Viterbi alignment of the text to the speech waveform for each of the training utterances. The canonical pronunciation of each word as defined by the lexicon, with the usual three-state, left-to-right model is used in this computation. The alignment step provides a labeled segmentation of each utterance into phonetic regions, with the labels corresponding to the states in the model for each phoneme.

The second phase of preprocessing is to perform some type of phoneme recognition on the training data. We have chosen to use the same alphabet as in the Viterbi alignment, i.e. thirds-of-phones. Our phoneme recognizer consists of the same acoustic models as the baseline speech recognition system along with a set of transition probabilities among context-dependent phones. Finding the most-likely sequence of phones yields quite clean label sequences. The resulting label sequences are segmented according to the subphonetic bound-

aries calculated in the Viterbi alignment and pooled according to the state label associated with the alignment.

The third and final phase of preprocessing is to partition the set of observation sequences for each phoneme into context-dependent units. For this we use a decision tree which asks questions about phonetic context and measures the goodness of a split in terms of the reduction in entropy in the distribution over labels. Each leaf in this tree represents a context-dependent unit for which we will build a pronunciation network using the technique outlined in the previous section. Dropping the training data down the tree yields a set of label sequences for each context-dependent unit from which to build the network which will characterize it.

Note that this tree is distinct from that which defines the context-dependent units for which acoustic models are built as will be discussed in the next section.

PRONUNCIATION MODELS GIVEN OBSERVED STRINGS

The discussion in this section assumes that we have a set of observations in the form of sequences of phonemic labels and that the observations have been partitioned based on phonetic context. These preprocessing steps were described in the previous section. Having obtained the observations, we would like to discover from them the observations a network of states which represents well the collection of observed label sequences for each context; the procedure by which we build a network to represent the observations for each context is the subject of this section.

This procedure is similar in spirit to that described in [Stolcke 94] but differs from their implementation in order to reduce the required computations.

For each context, we first discard all sequences containing a symbol which has not been observed at least N times in the pool of training sequences for this context, where N is a hand-set threshold. We then divide the remaining training observations into two sets, a "development" set for building the initial networks and a second "held-out" set for reducing the number of states in the network.

Next, we build a large initial network which can explain all the sequences in the development training set. We have investigated initializing this network as null and as the default three state left-to-right network and have found the latter to provide slightly better recognition results. Against the starting point of null or the default network, we check the first observation to see if it may be explained by the existing network. If so, we update transition probabilities only. If the observation does not align to the existing network, we add a parallel path consisting of one state each time the phoneme in the sequence differs from its left neighbor, with transition probabilities derived from the number of repetitions of the phonemes within the observation. For each remaining observation in the development training pool we repeat the procedure, checking whether it may be explained by the existing network; if so we update transition probabilities and if not we add a parallel path to the network. After all observations have been incorporated into the network, we prune all branches with transition probability into the branch less than ϵ where ϵ is a hand-determined threshold.

Having built the initial network, we begin collapsing states where such merges are favorable on our held-out training data. Each state in the network is labeled from the alphabet of subphonemic units, e.g. AA_1. Only states which carry the same label are considered for merge. The probability distribution over labels for each state is concentrated entirely on the associated label; the likelihood of the held-out data is computed before and after a given merge. If the likelihood increases, or if the number of observations which can be modeled by the resulting network is larger than was the case prior to the merge, the merge is retained; otherwise the two states are kept distinct. Because states are collapsed whenever an advantageous merge is found, the order in which states are evaluated for merging impacts the final network. We have somewhat arbitrarily started with the first two states having the same label and hold the first state fixed, sequentially evaluating the second state until a good merge is found. Once a merge is retained, we reset the starting state to be the next state in the network from the current starting state and iterate until no more good merges exist or until there are fewer than S states in the network, where S is a threshold set by hand.

Merging of two states consists of creating a new state whose parents are the union of the parents of the two states and whose children are itself plus the children of the two states excluding those states themselves (self-loops) and deleting the two unmerged states and their incoming and outgoing arcs.

Finally, two iterations of the E.M. algorithm are run to estimate transition probabilities for the network.

USING PRONUNCIATION NETWORKS IN RECOGNITION

During recognition, context-dependent pronunciation networks are used in lieu of the default three-

state model topologies. As the acoustic models are also context-dependent, two distinct decision trees are used by the recognizer: one to partition the space of observed label sequences based on phonetic contexts and one to partition the space of acoustic variations of each state in the network given the same phonetic context features. Having altered the phoneme topolgy one would expect that rebuilding of the decision trees to partition the acoustic variations for each state in the network is necessary. However, because the label alphabet for the states in the pronunciation networks is the same as the units the phonetic-context decision trees were built for in a baseline recognition system, we can simply cluster the states based on their labels and use the same decision trees as used in the baseline system, which used one tree for each label. The Gaussian mixture models for each leaf of these trees are those of the baseline system; no acoustic retraining is necessary. In fact, one disappointing result related to the pronunciation networks is that retraining the Gaussian model parameters using the E.M. algorithm and the pronunciation network topologies did not lead to a further decrease in the error.

DISCUSSION

The typical model topologies resulting from the procedure are described here for the case of training the networks on the Wall Street Journal data. In many cases (89% of the context-dependent units unweighted by frequency of occurrence) the network is exactly the default three-state left-to-right network. In 5% of the contexts the resulting network is the default network plus one skip arc. The next most common network has one additional state in parallel with the default. There is a relatively long tail, down to the case of a network with 14 states. The most common interchage of labels occurs in the networks associated with S and Z which is consistent with the result from speech science that the evidence to distinguish between these phonemes occurs close to the boundaries of the phone and that in the center of the region voicing activity for Z may not be apparent.

RESULTS

The error rates resulting from using the pronunciation networks to define the context-dependent phoneme model topologies are compared with the errors resulting from a baseline three-state, left-to-right model topology which has no skips. We report the error rates on both the Wall Street Journal and Broadcast News databases. On WSJ, the baseline of 8.0% error fell to 7.4% when the pronunciation network topologies were used in lieu of the default networks. On the Broadcast News data the error rate was again reduced by using the pronunciation networks, as detailed in table 1. Appealingly, the largest reduction in error occurs in the case of spontaneous speech, a condition where we would expect that pronunciations variations from the lexical representation might be larger than what would be observed in the case of planned speech, and therefore a condition for which this type of modeling would be potentially valuable.

	F0	F1	F2	F3	F4	F5	FX
A	9.1	20.8	28.0	25.1	24.4	19.6	37.1
B	8.9	20.1	27.8	25.0	24.4	19.5	37.4

Table 1. Error rates on Broadcast News data. Condition A is the baseline. Condition B uses pronunciation networks. F0=Clean/Planned Speech. F1=Clean Spontaneous Speech. F2=Speech on Telephone Channels. F3=Speech with Background Music. F4=Speech in Degraded Acoustics. F5=Non-native speakers. FX=Combinations of F1-F5.

Also, as the error rate on "fast" speakers in our baseline system is much higher than the error rate for other speakers, we have analyzed the performance gains obtained on the "fast" speakers and found the average error rate across all conditions to fall from 26.8 to 26.1 on those speakers by using the pronunciation networks. Fast speakers were identified subjectively by listening to the data as Donna Kelley and Leon Harris. The pronunciation networks are identical to those used for the results in table 1 and were trained on all the training data, not just that from fast speakers.

REFERENCES

Stolcke, A. and S. Omohundro. "Best-first Model Merging for Hidden Markov Model Induction." International Computer Science Institute report TR-94-003. January 1994.